```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</prop
erty>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/menu</prope
rty>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">asdf</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.format_sql">true</property>
        <mapping class="org.example.entity.MenuItem"/>
    </session-factory>
</hibernate-configuration>
```

MenuItem.java

```java
package org.example.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class MenuItem {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    private Double price;
    private String category;
    private Boolean available;

    public MenuItem(){}
    public MenuItem(String name, Double price, String category, Boolean
available) {
        this.name = name;
        this.price = price;
```

```
        this.category = category;
        this.available = available;
    }

    // get set for all var
    public int getId() { return id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public double getPrice() { return price; }
    public boolean getAvailable() { return available; }

    public void setPrice(double price) { this.price = price; }
    public String getCategory() { return category; }
    public void setCategory(String category) { this.category =
category; }
    public void setAvailable(boolean available) { this.available =
available; }

    @Override
    public String toString() {
        return id + " | " + name + " | " + price + " | " + category + "
| " + available;
    }


}
```

HibernateUtils.java

```
package org.example.HibernateUtils;

import org.example.entity.MenuItem;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;


public class HibernateUtils {

    private static final SessionFactory sessionFactory =
buildSessionFactory();

    private static SessionFactory buildSessionFactory(){
        try {
            return new Configuration()
                    .configure("hibernate.cfg.xml")
                    .addAnnotatedClass(MenuItem.class)
                    .buildSessionFactory();
```

```
        } catch (Exception e){
            throw new RuntimeException("[SessionFactory]" +
e.getMessage());
        }
    }


    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Main.java

```
package org.example;


import org.example.entity.MenuItem;
import org.hibernate.Session;
import org.hibernate.Transaction;


import java.util.List;
import java.util.Scanner;

import static
org.example.HibernateUtils.HibernateUtils.getSessionFactory;

public class App {
    private static Scanner scanner = new Scanner(System.in);
    public static void main( String[] args ) {
        boolean running = true;
        while(running) {
            System.out.println("\n\n1. Add Menu Item");
            System.out.println("2. View");
            System.out.println("3. Update");
            System.out.println("4. Delete");
            System.out.println("5. Exit");

            System.out.println("Enter choice: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1: addItem(); break;
                case 2: viewItem(); break;
                case 3: updateItems(); break;
                case 4: deleteItems(); break;
                case 5: running = false;
            }
```

```
        }
    }

    static void addItem(){
        Session session = getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();

        System.out.print("Name: ");
        String name = scanner.next();

        System.out.print("Price: ");
        Double price = scanner.nextDouble();

        System.out.print("Category: ");
        String category = scanner.next();

        System.out.println("Available [true/false]");
        Boolean available = scanner.nextBoolean();

        MenuItem menuItem = new MenuItem(name, price, category,
available);

        session.save(menuItem);
        tx.commit();
        session.close();
    }

    static void viewItem() {

        Session session = getSessionFactory().openSession();

        try {
            List<MenuItem> list = session.createQuery("from MenuItem",
MenuItem.class).list();

            for(MenuItem item : list) {
                System.out.println(
                        item.getId() + " | " +
                                item.getName() + " | " +
                                item.getPrice() + " | " +
                                item.getCategory() + " | " +
                                item.getAvailable()
                );
            }

        } finally {
            session.close();
        }
    }
```

```java
    static void updateItems() {

        Session session = getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();

        try {

            System.out.print("Enter ID to update: ");
            int id = scanner.nextInt();

            MenuItem item = session.get(MenuItem.class, id);

            if(item == null) {
                System.out.println("Item not found");
                return;
            }

            System.out.print("New Name: ");
            item.setName(scanner.next());

            System.out.print("New Price: ");
            item.setPrice(scanner.nextDouble());

            System.out.print("New Category: ");
            item.setCategory(scanner.next());

            System.out.print("Available [true/false]: ");
            item.setAvailable(scanner.nextBoolean());

            session.update(item);

            tx.commit();

        } catch(Exception e) {
            tx.rollback();
            e.printStackTrace();
        } finally {
            session.close();
        }
    }

    static void deleteItems() {

        Session session = getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();

        try {

            System.out.print("Enter ID to delete: ");
            int id = scanner.nextInt();
```

```
            MenuItem item = session.get(MenuItem.class, id);

            if(item == null) {
                System.out.println("Item not found");
                return;
            }

            session.delete(item);

            tx.commit();

            System.out.println("Deleted successfully.");

        } catch(Exception e) {
            tx.rollback();
            e.printStackTrace();
        } finally {
            session.close();
        }
    }

}
```

DB output:

	id	available	category	name	price
▶	2	1	camera	camera	10000
	3	1	LAptop	laptop	120000
	4	1	keyboard	keyborad	2000
	5	1	mouse	MOUSE	1200

Console Output:

```
2 | camera | 10000.0 | camera | true
3 | laptop | 120000.0 | LAptop | true
4 | keyborad | 2000.0 | keyboard | true
5 | MOUSE | 1200.0 | mouse | true


1. Add Menu Item
2. View
3. Update
4. Delete
5. Exit
Enter choice:
```